Theoretical Computer Science 62 (1988) 319-325
North-Holland

NOTE

THE POWER OF NONDETERMINISM IN POLYNOMIAL-SIZE BOUNDED-WIDTH BRANCHING PROGRAMS

Christoph MEINEL

Karl-Weierstrass-Institut für Mathematik, Akademie der Wissenschaften, DDR-1086 Berlin, PF 1304, German Dem. Rep.

Communicated by M.S. Paterson
Received April 1987
Revised January 1988

Abstract. Nondeterministic branching programs introduced by Meinel (1986) proved to be an interesting computational tool for describing higher complexity classes (Meinel 1988). Investigation of the power of nondeterminism in the case of bounded-width nondeterministic branching programs yields the following results:

(i) 1-time-only-nondeterministic, polynomial-size, and bounded-width branching programs are no more powerful than (ordinary) polynomial-size and bounded-width branching programs;

(ii) k-times-only-nondeterministic, polynomial-size, and bounded-width branching programs, ($k>1$), are as powerful as nondeterministic, polynomial-size, and unbounded-width branching programs.

That is,

$$\mathcal{P}_{\text{bw-}n_1\text{BP}} = \text{NC}^1 \quad \text{and} \quad \mathcal{P}_{\text{bw-}n_k\text{BP}} = \text{NP/poly}, \quad k>1.$$

Introduction

One of the major goals in complexity theory is to separate complexity classes such as L, NL, P and NP (or to prove their coincidence) or, equivalently, to show that nondeterministic Turing machines with certain resource restrictions are more powerful than deterministic ones (or not). Since combinatorial techniques and counting arguments, which are expected to be of fundamental importance in doing this, can be applied more directly to circuit-based computation devices such as Boolean circuits or branching programs than to complex types of Turing machines, circuit-based characterizations of complexity classes gain more and more importance. For example, it is known that

- the class L/poly of functions nonuniformly computable by log-space bounded Turing machines can be described in terms of polynomial-size branching programs;
- the nondeterministic counterpart NL/poly of L/poly can be described by means of polynomial-size, 1-time-only-nondeterministic branching programs [4];

- the class NP/poly of functions nonuniformly computable by nondeterministic polynomial-time bounded Turing machines consists of all functions computable by sequences of nondeterministic polynomial-size branching programs [5]; and
- the class NC^1 of all functions computable by sequences of Boolean circuits with fan-in 2 and depth $O(\log n)$ can be characterized by polynomial-size, bounded-width branching programs [1].

By means of these branching program characterizations, questions concerning the separation of complexity classes can be formulated as questions about the power of such branching program devices. In the following we investigate nondeterministic bounded-width branching programs in order to understand better the power of nondeterminism in such programs. It will be proved that 1-time-only-nondeterministic, polynomial-size, and bounded-width branching programs are no more powerful than deterministic ones, while k-times-only-nondeterministic, polynomial-size, and bounded-width branching programs are as powerful as nondeterministic, polynomial-size, unbounded-width branching programs for $k > 1$.

1. Definitions and notations

As usual we denote by L (NL and NP respectively) the complexity classes of languages $A \subseteq \{0,1\}^*$ which are acceptable by *deterministic* $(\log n)$*-space bounded* (*nondeterministic* $(\log n)$*-space bounded*, and *nondeterminstic polynomial-time bounded*) Turing machines M. The nonuniform counterparts L/poly, NL/poly, and NP/poly are the classes of languages $A \subseteq \{0,1\}^*$ for which there is an *advice* $\alpha : \mathbb{N} \to \{0,1\}^*$, length-bounded by a polynomial p, i.e., $|\alpha(n)| < p(n)$, such that M *accepts* $a \# \alpha(|a|)$ if $a \in A$ ($\#$ stands for the blank tape symbol) [3]. Further, NC^1 denotes the (nonuniform) complexity class of languages $A \subseteq \{0,1\}^*$ whose restrictions $A^n = A \cap \{0,1\}^n$ can be computed by (polynomial-size) fan-in 2 Boolean circuits of depth $O(\log n)$.

A *branching program* P is a directed acyclic graph where each node has outdegree 2 or 0. Nodes with outdegree 0 are called *sinks* and are labelled by Boolean constants. The remaining nodes are labelled by Boolean variables taken from a set $X = \{x_1, \ldots, x_n\}$. There is a distinguished node, called the *source*, which has indegree 0. A branching program *computes* an n-argument Boolean function as follows: starting at the source, the value of the variable labelling the current node is tested. If it is 0 (1), the next node tested is the left (right) descendant of the current node. P *accepts* $A \subseteq \{0,1\}^n$ if for all $a \in \{0,1\}^n$ the path traced from the source under a halts at a sink labelled by $\chi_A(a)$ where χ_A denotes the characteristic function of A. The *complexity measure* for a branching program is the number of non-sink nodes. If $\mathcal{P}_{BP}$ denotes the class of all languages $A \subseteq \{0,1\}^*$ whose restrictions A^n will be accepted by polynomial-size branching programs, then it holds that

$$L/poly = \mathcal{P}_{BP}$$

by means of a classical result of Savitch [6].

In order to describe higher complexity classes by means of branching programs, nondeterministic branching programs were introduced [4]. A branching program P is called a *nondeterministic branching program accepting* A^n if there is a function

$$h:\{0,1\}^{n+m}\to\{0,1\},\quad m>0$$

with

$$\chi_{A^n}(x_1,\ldots,x_n)=\bigvee_{y\in\{0,1\}^m} h(x_1,\ldots,x_n,y)$$

and if P is a branching program computing h. P is called *k-times-only-nondeterministic* if each of the (nondeterministic) variables y_i on any path of P from the source to a sink is tested at most k times. If $\mathcal{P}_{nBP}$ and $\mathcal{P}_{n_kBP}$, $k\in\mathbb{N}$, denote the classes of all languages A whose restrictions A^n will be accepted by polynomial-size nondeterministic branching programs, or k-times-only-nondeterministic branching programs respectively, then the following hold [4, 5]:

$$\mathrm{NL/poly}=\mathcal{P}_{n_1BP}\quad\text{and}\quad\mathrm{NP/poly}=\mathcal{P}_{nBP}.$$

Finally, Barrington succeeded in describing NC^1 by means of polynomial-size, bounded-width branching programs. According to [2], we define the *width* of a branching program as the maximal number, over all d, of vertices at distance d from the source. Thereby, the *distance* $d(v)$ is the length of a longest path from the source to the vertex v. (Note that in this definition the underlying graph of the branching program is not required to be levelled.) If $\mathcal{P}_{\text{bw-BP}}$ denotes the class of all languages $A\subseteq\{0,1\}^*$ whose restrictions A^n will be accepted by polynomial-size branching programs of bounded width, then [1]

$$\mathrm{NC}^1=\mathcal{P}_{\text{bw-BP}}.$$

Moreover, Barrington proved $\mathrm{NC}^1=\mathcal{P}_{\text{width-}k\text{ BP}}$ for all $k\geq 5$, where $\mathcal{P}_{\text{width-}k\text{ BP}}$ denotes the class of all languages whose restrictions will be accepted by polynomial-size branching programs of width k.

2. Results

Theorem 1. $\mathcal{P}_{\text{bw-}n_1\text{BP}}=\mathrm{NC}^1$.

Proof. As a consequence of [1], it suffices to prove that $\mathcal{P}_{\text{bw-BP}}=\mathcal{P}_{\text{bw-}n_1\text{BP}}$.

Let $A\in\mathcal{P}_{\text{bw-}n_1\text{BP}}$. Then there exists a $k\in\mathbb{N}$ such that every restriction A^n of A will be accepted by a 1-time-only-nondeterministic polynomial-size branching program P' of width k'. Since two 1-time-only-nondeterministic branching programs which differ only in the nondeterministic variables assigned to the nodes compute the same function, we can assume that every nondeterministic variable of P' is assigned exactly once to a node of P'. Thus, by means of the construction proposed in [1], we can transform P' into a leveled 1-time-only-nondeterministic polynomial-size

branching program P of width k, $k<2k'$, all of whose nodes on the same level access the same (deterministic or nondeterministic) input variable, and which accepts A^n. Obviously, a level j of P labelled by the variable z_i is completely described by two functions

$$f_j, g_j : [k] \to [k],$$

where $[k] := \{1, \ldots, k\}$. $f_j(v)$ and $g_j(v)$ give the end points in level $j+1$ of the two edges leaving a node $v \in [k]$ of level j for $z_i = 0$ and $z_i = 1$ respectively.

Now we simulate the 1-time-only-nondeterministic branching program P level by level by an (ordinary) branching program P'' of width 2^k whose length equals that of P, which implies that $A \in \mathcal{P}_{\text{bw-BP}}$. In order to do this, identify the nodes of each level of P'' with the elements M of $2^{[k]}$. If level j of P is labelled by the deterministic variable x_i, then label the nodes of level j of P'' by x_i too, and define for $M \in 2^{[k]}$

$$f_j, g_j : 2^{[k]} \to 2^{[k]}$$

by

$$f_j(M) = \{f_j(m) \mid m \in M\} \quad \text{and} \quad g_j(M) = \{g_j(m) \mid m \in M\}.$$

But if level j of P is labelled by the nondeterministic variable y_i, then label the nodes of level j of P'' by any one of the deterministic variables $x_1, \ldots, x_n$ and define

$$f_j(M) = g_j(M) = \bigcup_{m \in M} \{f_j(m)\} \cup \{g_j(m)\}.$$

Obviously, P is of polynomial size and accepts exactly A^n.

Conversely, since every bounded-width branching program is a 1-time-only-nondeterministic one of bounded width, we are finished. □

Theorem 2. $\mathcal{P}_{\text{bw-}n_2\text{BP}} = \text{NP/poly}$.

Proof. It is well known that NP/poly coincides with the class of all languages that are computable by sequences of polynomial-size, *nondeterministic circuits*. A $\{\vee, \wedge\}$-circuit C whose input nodes are labelled by Boolean constants and Boolean literals over the set $\{x_1, \ldots, x_n, y_1, \ldots, y_m\}$ is said to compute a set $A^n \subseteq \{0,1\}^n$ *nondeterministically* if for all $a \in A^n$ there is an assignment α of Boolean constants to the nondeterministic variables $y_1, \ldots, y_m$ such that C outputs 1, and if for all $a \notin A^n$ and for all such α, C outputs 0. Equivalently

$$\chi_{A^n}(a) = \bigvee_{\alpha \in \{0,1\}^m} C(a, \alpha).$$

Therefore it suffices to prove that $\mathcal{P}_{\text{bw-}n_2\text{BP}}$ coincides with $\mathcal{P}_{n\text{Cir}}$, the class of all languages computable by sequences of nondeterministic polynomial-size circuits.

The assertion that $\mathcal{P}_{\text{bw-}n_2\text{BP}} = \mathcal{P}_{n\text{Cir}}$ is a consequence of the following four lemmas.

Lemma 1. $\mathcal{P}_{\text{bw-}n\text{BP}} \subseteq \mathcal{P}_{n\text{Cir}}$.

Proof. Adapting a construction for deterministic branching programs [8], from a nondeterministic branching program P one obtains a nondeterministic {sel}-circuit C_P computing the same set as P. The function sel is defined by

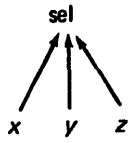
$$\text{sel}(x, y, z) = \bar{x} \wedge y \vee x \wedge z \quad \text{for all } x, y, z \in \{0, 1\}.$$

C_P is constructed from P by reversing the directions of all edges of P, labelling each node v of P with sel and providing it with a new predecessor, namely the circuit input node of the variable z_i by which the point v is labelled in P. The descendant of v which was reached in P if $z_i = 0$ is taken as the second predecessor and the descendant which was reached if $z_i = 1$ is taken as the third. Replacing all sel-nodes in C_P by the 3-node subcircuit C_{sel} as shown in Fig. 1, one obtains a $\{\vee, \wedge\}$-circuit C'_P with

$$\text{Size}(C'_P) = O(\text{Size}(C_P)) = O(\text{Size}(P))$$

which computes the same set as C_P. This implies $\mathcal{P}_{\text{bw-}n\text{BP}} \subseteq \mathcal{P}_{n\text{Cir}}$. □

Let $\mathcal{P}_{\text{depth-2},n\text{Cir}}$ denote the class of all sets computable by sequences of nondeterministic, unrestricted fan-in, polynomial-size $\{\vee, \wedge\}$-circuits of depth 2; then we have the following lemma.



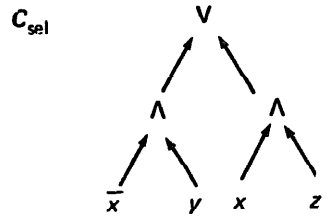


Fig. 1.

Lemma 2. $\mathcal{P}_{n\text{Cir}} \subseteq \mathcal{P}_{\text{depth-2},n\text{Cir}}$.

Proof. Let C be a nondeterministic circuit computing a set A^n. From C one obtains a depth-2, nondeterministic circuit C' computing A^n in the following way. First we assign a Boolean variable $\alpha(v)$ to each node v of C such that

(i) if v is an internal node, we assign to v a nondeterministic variable $y_v = \alpha(v)$, and

(ii) if v is an input node of C, then we assign to v the deterministic or nondeterministic variable with which v is labelled.

Then we compute at a node v with predecessors $v_1, \ldots, v_r$ the Boolean equivalence

$$y_v \leftrightarrow \alpha(v_1) \wedge \cdots \wedge \alpha(v_r) \quad \text{if } v \text{ is an } \wedge\text{-node,}$$

or

$$y_v \leftrightarrow \alpha(v_1) \vee \cdots \vee \alpha(v_r) \quad \text{if } v \text{ is an } \vee\text{-node.}$$

Together with the final check whether all equivalences hold, these computations can be done in parallel by a nondeterministic circuit of depth 2. □

Lemma 3. $\mathcal{P}_{\text{depth-2},n\text{Cir}} \subseteq \mathcal{P}_{\text{width-2},n\text{BP}}$.

Proof. Again we can adapt a construction of Wegener [8] for getting width-2 branching programs from depth-2 circuits.

Obviously, the nondeterministic width-2 branching programs P_c and P_d shown in Fig. 2 compute the conjunction $c = x_1 \wedge x_2 \wedge \cdots \wedge x_r$ and the disjunction $d = x_1 \vee x_2 \vee \cdots \vee x_r$ respectively.

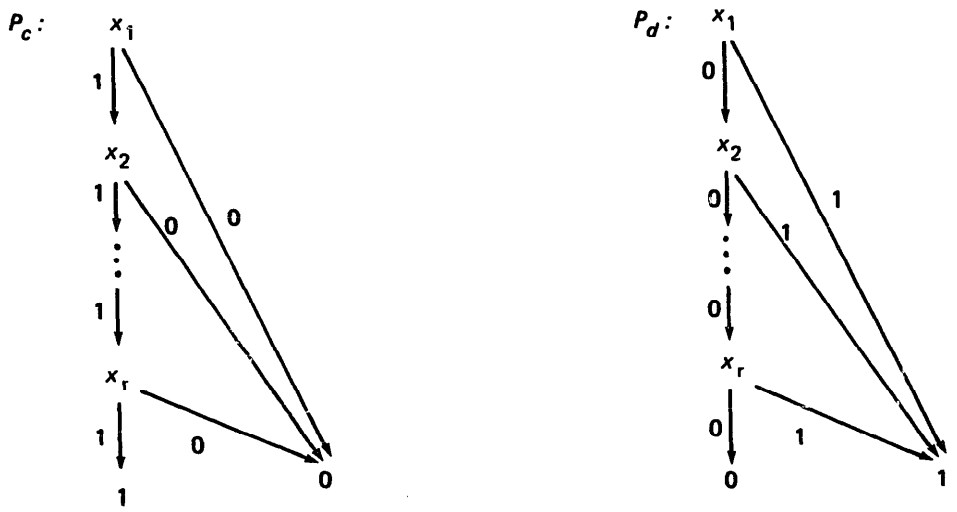


Fig. 2.

Let C be a nondeterministic depth-2 circuit whose least gate is a conjunction (similar arguments work for a disjunction). If $d_1, \ldots, d_m$ are the disjunctions computed in the first logical level of C, then we can simulate C by a nondeterministic branching program P_C constructed from the nondeterministic branching programs $P_{d_1}, \ldots, P_{d_m}$ by identifying the 1-sink of P_{d_i} with the starting node of $P_{d_{i+1}}$ for each i, $1 \leq i < m$. Since P_C is of width 2 and of polynomial size in the number of gates of C we are done. □

Remark. If one requires the branching programs under consideration to be levelled, then the construction of Lemma 3 provides a nondeterministic branching program of width 3.

Lemma 4. $\mathcal{P}_{\text{width-}k,n\text{BP}} \subseteq \mathcal{P}_{\text{width-}k,n_2\text{BP}}$, $k \geq 2$.

Proof. Let P be a nondeterministic width-k branching program. We first obtain a 1-time-only-nondeterminstic branching program P' of width k by replacing the

variables z_i assigned to nodes v of P by new nondeterministic variables $z_{i,v}$. We then get a 2-times-only-nondeterministic branching program P'' accepting the same set as P, if we also check all Boolean equivalences

$$\left(\bigwedge_{v \in P} z_{i,v}\right) \leftrightarrow z_i$$

for all nondeterministic variables z_i before accepting an input. Since these computations can be done by polynomial-size depth-2 circuits we can, due to Lemma 3, check these equivalences by polynomial-size branching programs of width 2. □

Proof of Theorem 2 (*conclusion*). Indeed, due to Lemma 4, we have proved that 2-times-only-nondeterministic branching programs of width 2 are as powerful as nondeterministic polynomial-size circuits, which sharpens a result of Valiant [7]. □

References

[1] D.A. Barrington, Bounded-width polynomial-size branching programs recognizing exactly those languages in NC^1, in: *Proc. 18th Symp. on Theory of Computing* (1986) 1-5.

[2] A. Borodin, D. Dolev, F.E. Fich and W. Paul, Bounds for width-2 branching programs, in: *Proc. 15th Symp. on Theory of Computing* (1983) 87-93.

[3] R.M. Karp, R.J. Lipton, Some connections between nonuniform and uniform complexity classes, in: *Proc. 12th Symp. on Theory of Computing* (1980) 302-309.

[4] Ch. Meinel, p-Projection reducibility and the complexity classes L (nonuniform) and NL (nonuniform), in: *Proc. 12th Conf. on Mathematical Foundations of Computer Science*, Bratislava (1986) 527-535.

[5] Ch. Meinel, Modified branching programs and their computational power, Habilitationsschrift (Berlin, 1988).

[6] W. Savitch, Relations between nondeterministic and deterministic tape complexities, *J. Comput. System Sci.* 4 (1970) 177-192.

[7] L. Valiant, Reducibility by algebraic projections, *Enseign. Math.* XXVIII (3-4) (1981) 253-268.

[8] I. Wegener, *The Complexity of Boolean Functions*, Wiley-Teubner Series in Computer Sciences (Stuttgart, 1987).